Energy Proportional Streaming Spiking Neural Network in a Reconfigurable System

FELIPE GALINDO SANCHEZ, University of Bristol
JOSE NUNEZ-YANEZ, University of Bristol

This paper presents a high-performance architecture for spiking neural networks that optimizes data precision and streaming of configuration data stored in main memory. The neural network is based on the Izhikevich model and mapped to a CPU-FPGA hybrid device using a high-level synthesis flow. The active area of the network is configurable and this feature is used to create an energy proportional system. Voltage and frequency scaling are applied to the processing hardware and memory system to deliver enough processing and memory bandwidth to maintain real-time performance at minimum power and energy levels. The experiments show that the application of voltage and frequency scaling to DDR memory and programmable logic can reduce its energy requirements by up to 77% and 76% respectively. The performance evaluation show that the solution is superior to competing high-performance hardware, while voltage and frequency scaling reduces overall energy requirements to less than 2% of a software-only implementation at the same level of performance.

Index terms: Hardware → Integrated circuits → Reconfigurable logic and FPGAs → Hardware accelerators, Hardware → Integrated circuits → Reconfigurable logic and FPGAs → Reconfigurable logic applications

1. INTRODUCTION

The ability to understand parts of the human brain and being able to perform large-scale simulations has allowed biologically inspired techniques in applications such as speech recognition, computer vision, natural language processing, drug discovery and pattern recognition, among others. The high computational costs of this processing has resulted in the development of high-performance solutions describing biologically realistic neuron models in hardware accelerated solutions using Field Programmable Gate Array (FPGAs) and Graphical Processing Units (GPUs).

In this paper we propose a fully connected feed-forward network using the Izhikevich's neuron model combined with a synapse model in which the post-synaptic current is modelled as a function of the synaptic conductance and the differential of the membrane potential and the reversal potential. Both models are described in C++ and then targeted to a Zynq MPSoC using high-level synthesis and implementation tools. The hardware uses an optimized synaptic weight representation and transmission using the AXI4-Stream as the primary data protocol. The current solution shows excellent scalable properties with configurations up to 250K neurons and 125M synapses possible in a single Zynq 7100 device. The performance evaluated on a Zynq 7020 device with the simulation of 28,900 neurons and 5M synapses results in speedups of 5 to 7 times in contrast with alternative computing solutions while using less than 2% of the energy of an Intel solution based on OpenCL.

This work is supported by the UK EPSRC under grants ENPOWER and ENEAC

Author's addresses: Felipe Galindo Sanchez, University of Bristol, Woodland Road, Bristol, BS8 1UB. j
and Jose Nunez-Yanez, Electronic Engineering Department, University of Bristol, Woodland Road, Bristol, BS8 1UB. j

The research contributions of this work can be summarized as follows:

- We create the first high-performance configurable system for spiking neural networks (SNN) targeting the Zynq MPSoC devices based on a C++ high level synthesis flow. The hardware allows multiple configuration options including data precision, network size and host connectivity. The solution uses an optimized streaming strategy to reduce the pressured on internal BlockRAM resources and it does not require any RTL coding.
- To the best of our knowledge we apply for the first time energy proportional concepts based on dynamic voltage and frequency scaling (DVFS) to both programmable logic and memory system in the SNN demonstrating its effectiveness in a practical scenario and under real-time performance constraints.
- We release the streaming spiking neural network named S2NN as an open-source downloadable package in synthesizable C++ and also an equivalent OpenCL description used as a benchmark to motivate further research in the area and to create reproducible research. To the best of our knowledge this has not been available before.

2. BACKGROUND AND RELATED WORK

2.1 Background

The selected Izhikevich neuronal model combines the biologically plausibility of the Hodgkin-Huxley's model and the computational flexibility of the integrate and fire model [Izhikevich 2004]. This model is able to represent different spiking and bursting neuron behaviors with only two equations involving four independent model parameters: a,b c and d, one recovery variable u(t) and one non-linear term $\mathrm{v}(t)^2$ as shown in Equation (1).

$$\mathrm{v}(t)' = 0.04\mathrm{v}(t)^2 + 5\mathrm{v}(t) + 140 - \mathrm{u}(t) + \mathrm{I_{syn}}(t) \tag{1}$$

$$\mathrm{u}(t)' = \mathrm{a} \cdot [\mathrm{b} \cdot \mathrm{v}(t) - \mathrm{u}(t)]$$

$$\text{if } \mathrm{v}(t) \geq 30\mathrm{mV} \rightarrow \begin{cases} \mathrm{v}(t+1) = \mathrm{c} \\ \mathrm{u}(t+1) = \mathrm{u}(t) + \mathrm{d} \end{cases}$$

As previously mentioned, this neuron model is then combined with a synapse model in which the post-synaptic current $I_{syn}(t)$ is modelled as a function of the synaptic conductance $g_{syn}(t)$ and the differential of the membrane potential $v(t)$ and the reversal potential E_{syn} as shown in Equation (2). For excitatory synapses the typical reversal potential is $E_{syn} = -75mV$ whereas for inhibitory synapses it is $E_{syn} = 0$; hence the behavior of increasing or decreasing the membrane potential of the neuron for excitatory and inhibitory synapses respectively can be inferred by the sign of the differential of both potentials $v(t) - E_{syn}$.

$$I_{syn}(t) = g_{syn}(t) \cdot \big(v(t) - E_{syn}\big) \tag{2}$$

The synaptic current $I_{syn}(t)$ value for a N number of synapses connected to a neuron, is described in the following Equation (3) as the sum of the multiple synapses with synapses weights w and reversal potentials. The values of the synapses weights reflect the strengths of the connection and define the behavior of the network after training.

$$g_{syn}(t) = w \cdot s(t)$$

$$I_{syn}(t) = \sum_j w_{jk} \cdot s_k(t) \cdot \left(v_k(t) - E_j\right) \quad (3)$$

The synapse $s(t)$ value is modelled as an alpha function where the rising part refers to the binding of the neurotransmitter of the pre-synaptic neuron to the different gating channels; while the falling part refers to the unbinding from the cleft at a certain rate modeled by the synaptic time constant τ_s as in Equation 4.

$$s(t) = te^{-t/\tau_s} \quad (4)$$

Because of the high-cost implementation of exponential functions in a computational system, the falling part of the synaptic conductance $s(t)$ is described as a proportional decay with a synaptic time constant expressed in Equation 5, and the rising part can be modelled as an increase whenever a spike arrives to the neuron.

$$\tau_s \frac{ds}{dt} = -s(t) \text{ with } s(t) \rightarrow s(t) + 1 \text{ whenever there is a spike} \quad (5)$$

These equations are well known and have been used in related research activities as discussed in Section 2.2. The topology deployed in this research is a feed-forward topology consisting of multiple layers of neurons, where neurons in a certain layer are connected to the neurons of the incoming layer, and hence, the information flows in one direction, from the input layer to the output layer. Although this topology is classic, the novelty of this research is the effectiveness and performance of the implementation using a high-level synthesis flow based on C++ code without recurring to low-level RTL design and the exploitation for the first time of energy proportional principles at the reconfigurable and memory levels in order to adjust the level of energy and power required to the size of the network. The idea of energy proportional computing refers to being able to develop machines that consume energy in proportion to the amount of work performed [Barroso et al. 2007] and it is particularly applicable to systems that always need to maintain a certain level of activity that prevents them from entering full shutdown modes for significant periods of time. This will be the application scenario if this network is used to perform cognitive tasks in an energy constrained environment such as autonomous vehicle navigation or automatic speech recognition in robots.

Previous efforts in this area have succesfully applied techniques such as clock gating, power gating and dynamic voltage frequency scaling to the processors, and more recently to FPGAs [Hosseinabady et al. 2015]. In this work we focus on the FPGA and main memory that are the main components involved in spiking neural network processing, while the CPU enters into a sleep mode waiting for the interrupt that indicates that the processing of the time-step has completed. We extend this scaling to the main memory system that, as we will see, is the highest contributor to overall

power comsumption. The experimental evaluation with real hardware of voltage and frequency scaling applied to the main memory system complements and confirms the simulation-based study done in [David et al. 2011]. This previous research showed the possible benefits of DVFS applied to memory via an analytical model of memory power, while our research measures memory power at different experimental levels of performance defined by voltage-frequency pairs.

2.2 Related work

The area of neuromorphic computing has been receiving a lot of attention with major projects led by both industry and academia. For example, an industrial program led by IBM has recently deployed the TrueNorth brain-inspired neuro processor. TrueNorth is a massively parallel device with 4096 cores per chip capable of simulating complex recurrent neural networks. The next objective of the TrueNorth team is the integration of 4,096 chips in a single rack with 4 billion neurons and 1 trillion synapses while consuming ~4kW of power [Akopyan et al 2015]. At the academic level, the human brain project (HBP, www.humanbrainproject.eu) deserves also a special mention. HBP is funded by the EU's ICT program with a projected cost of one billion euros and includes several themes with the objective of understanding how the human brain operates. The neuromorphic theme develops innovative hardware architectures such as the Spinnaker chip [Furber et al.] that consists of a Globally Asynchronous Locally Synchronous (GALS) system with 18 ARM968 processor nodes. In addition to these large projects aim of full brain modelling, neural networks in embedded systems such as the Zynq family can play a role in lower cost applications such as automotive and robotics which is the objective of this work. Neural networks in this field are created with model complexity that ranges from the simple Integrate and Fire to highly complex descriptions such as Hodgkin–Huxley's. An intermediate complexity model such as Izhikevich's is considered to be a good trade-off with a low cost implementation point while maintaining a good number of biological features; consequently, this model is the first choice in a significant number of research efforts working with different design languages with significant examples described below.

[K. Cheung et al. 2009] and [M. Ambroise et al. 2013] choose to design their networks using a register transfer level (RTL) language such as VHDL. Although this has the benefit of fine control over the implementations details, it significantly reduces the overall design productivity compared with a higher-level abstraction flow. The proposed networks are small with 800 and 117 neurons respectively. [K. Rice et al. 2009] and [M. Bhuiyan et al. 2010] proposes a shallow two-layer network for a character recognition application resulting in a configuration with a limited number of synapses. The system is prototype in the SRC 7 H MAP which is a large scale system combining Xeon processors and several Altera Stratix FPGAs with access to a large pool of memory resources. The problem of how to scale the interconnect to be able to build large neuronal networks has been address in [Vainbrand et al. 2011]. The paper shows that multicast mesh NoC provides the highest performance/cost ratio and consequently it is a suitable interconnect architecture for configurable neural network implementation.

In addition to the neural model selected, the implementation data precision is also critical; [K. Rice et al. 2009] chooses a fixed-point implementation to reduce the logic resource footprint over a floating-point and a lowest representation of 12-bits ensuring that results do not produce incorrect character recognition. [M. Bhuiyan et al. 2010]

implements also a character recognition using Izhikevich's model but it also uses the more complex Hodgkin-Huxley's with six different implementation techniques of which only four fit in the FPGA due to the 265 FLOPs that each neuron requires to compute an update. Complex implementations using variants of Hodgkin-Huxley's model are presented also in [G. Smaragdos et al. 2014] and [Moctezuma et al. 2015] and due to the complex arithmetic that involves large quantities of exponentials and floating point operations the networks do not go above hundreds of neurons. In contrast, [F. Naveros et al. 2015] chooses a simpler model targetting a GPU/CPU architecture based on a leaky variant of the Integrate and Fire model, and thanks to its simpler implementation, larger networks are able to be simulated. The proposed time-driven GPU technique and event-driven CPU technique is able to simulate up to 50 000 neurons and 4.13 million synapses in real time using Nvidia GTX 470 and Intel Core I7 CPUs. In contrast our embeded system is much more compact and suitable for battery-powered applications. Storage and memory resource optimization are essential considerations and often the aim is to maximize their utilization. That is the case in [K. Cheung et al. 2009] that uses 18-bit and 9-bit two's complement precision to fully utilise the 18-bit two's complement multipliers of the DSP blocks available in the FPGA device. Another example is driven by memory availability; in this case [D. Thomas et al. 2009] stores four synaptic weights of 9 bits in one 36-bit register in internal BRAM to maximize BRAM utilization using C as the high-level design language. Final benchmarks comparing such implementations using high-end processors, FPGAs and GPUs show that the FPGA implementation results in 100 times faster than real-time for a fully-connected network with 1024 neurons. Real-time in this case refers to the value of 1 ms for simulation time steps so that enough accuracy is maintained for the representation of the spikes [Izhikevich, 2004]. If the hardware can compute one time step in less than 1ms then it is said to be real-time. [K. Cheung et al. 2012] uses a Maxeler high performance computing platform and Java as its design language and distributes synaptic data through a weight distribution controller that manages the information of synapses in 32-bits including synaptic weight, index of neuron and an axonal delay. [S. Moore et al. 2012] applies also the Izhikevich's equations with a 16-bit fixed-point precision. This system uses a "communication-centric" approach through a custom-made PCIe-to-SATA adapter that allows to interconnect up to 16 Altera DE4 FPGA boards with simulations up to 64K neurons and 64M synapses. The language of choice is Bluespec, a high-level functional hardware description programming language based on SystemVerilog, An example of a modular spiking network based on a network-on-chip interconnect is shown in [Pande et al. 2013]. The modular architecture is based on a LIF neuron which occupies 12 slices per neuron with a basic module of 16x16 or 256 neurons. The device is targetted to a Virtex-6 FPGA and can be scaled to multiple devices thanks the to NoC infrastructure.

A summary of this related work is shown in Table II while Table I clarifies the acronyms used in Table II. Some of the related work also utilize a high-level design language but in contrast our work does not use a high-performance computing platform but focuses on a combined optimization of data precision and data streaming to be able to fit the whole network in a small embedded FPGA-CPU device with a large utilization factor of all the resources. The presented research in this paper also shows how the system can be made energy proportional by selectively activating network areas and deploying voltage and frequency scaling at the computing and memory levels. These means that in contrast to large scale implementations that make use of PCIe boards controlled by desktop processors such as the SRC 7 H MAP [M. Bhuiyan et al. 2010] and the Maxeler boards used in [K. Cheung et al. 2012] our compact solution

could be used in energy constraint scenarios such as battery-powered embedded systems and robotics.

Table I. Related work acronyms

Neuron Models:
IZH - Izhikevich, ***HH*** - Hodgkin-Huxley, ***LIF*** - Leaky Integrated and Fire, ***2PR*** - Two-compartment P–R
Data Precision:
FLO - Floating Point, ***FIX*** - Fixed Point
Network Type:
FFW - Feed Forward, ***FCN*** - Fully connected, ***2LA*** - Two layers, ***3LA*** - Three layers
Framework/Language:
C - C HLS, ***JAV*** - Java HLS, ***VHD*** - VHDL, ***BSV*** - Bluespec SystemVerilog,
OMP - OpenMP, ***VIV*** - Vivado HLS tools, ***CUD*** - CUDA (Compatible with C, C++, Fortran),
HWN - Hardware description architecture (language no specified),
CAR - Carte programming environment (Compatible with C and Fortran),
EDL - EDLUT application for neural networks simulations (CUDA compatible)

Table II. Related work summary

Ref.	*Neuron Models*	*Data Precision*	*Network Type*	*Network Size*	*FPGA Family*	*FPGA Freq.*	*LUTs Used*	*GPU Family*	*Framework /Language*
K. Cheung et al. 2009	IZH	FIX	FCN	800	Virtex 5	110 MHz	35 K	-	VHD
M. Ambroise et al. 2013	IZH	FLO	FCN	117	Virtex 4	85 MHz	1 K	-	HWN
K. Rice et al. 2009	IZH	FIX	2LA	1K	Virtex 4	198 MHz	N/A	-	C
M. Bhuiyan et al. 2010	IZH, HH	FLO	2LA	2 M	Altera II	150 MHz	N/A	-	CAR
F. Naveros et al. 2015	LIF	FLO	3LA	100 K	-	-	-	NVIDIA	OMP, EDL
D. Thomas et al. 2009	IZH	FLO	FCN	1K	Virtex 5	133 MHz	27 K	-	C, CUD
K. Cheung et al. 2012	IZH	FIX	N/A	64K	Virtex 6	100 MHz	205 K	-	JAV
S. Moore et al. 2012	IZH	FIX	N/A	64K	Altera II	200 MHz	N/A	-	BSV
G. Smaragdos et al. 2014	ION/HH	FLO	FCN	96	Virtex 7	100 MHz	251 K	-	C, VIV
Y. Zhang et al. 2009	2PR	FLO	FCN	105	Virtex 6	100 MHz	108K	-	VHD
Pande et al. 2013	LIF	FIX	FCN	256	Virtex 6	200 MHz	4816		VHD

3. SYSTEM DESCRIPTION

The network topology shown in Fig. 1 is defined as a $L\ x\ n_{layer}$ network configuration, where every layer can have a maximum number of n_{layer} neurons, and every neuron at most s_{neuron} synapses. In the considered fully connected configuration each neurons connects to all the neurons in the previous layer so $n_{layer} = s_{neuron}$. The three main processing blocks that need to be executed every simulation step of 1 ms are described in Fig. 2.

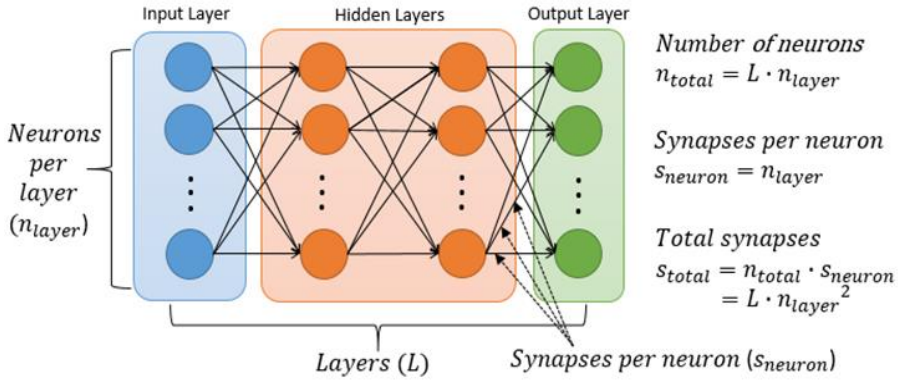

Fig. 1. SNN Network topology

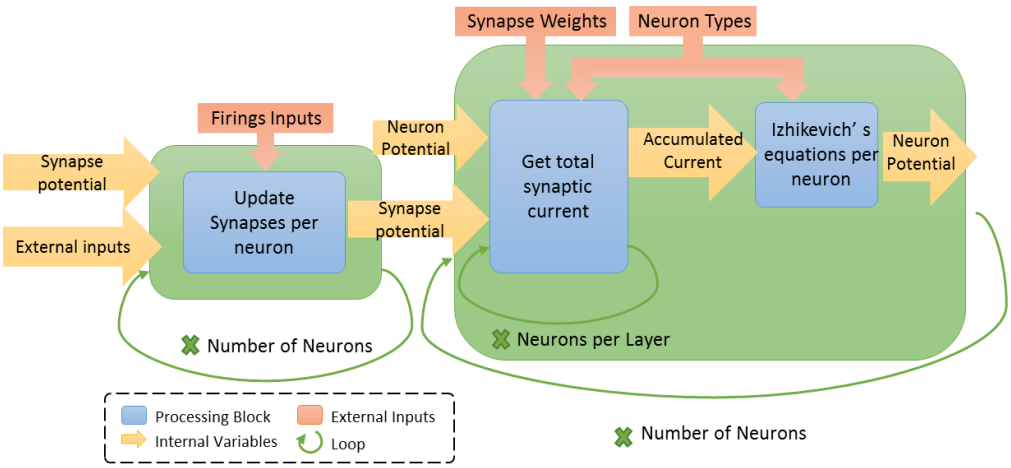

Fig. 2. Processing blocks, variables and loops

Table III shows the result of profiling the network based on a C implementation and an Intel x86 architecture. The table shows that the bottleneck resides in the second processing block where the number of iterations is n_{layer} followed by the neuron equations itself. The accumulated current I_{syn} value is the input into the neuron model and therefore both blocks are tightly integrated and can be mapped to hardware as a single unit. The description of the neural network in C++ was synthesized to RTL using Vivado HLS. One of the key resources that must be optimized is Block RAM memory utilization. The size of all arrays involved in the proposed algorithm is proportional to the number of neurons n_{total} with the exception of the synapse weights

proportional to $n_{total}\ x\ n_{layer}$. In the proposed architecture there are two parameters that are required per neuron in a *NxN* network:

- Neuron type (1 bit): N^3 values = N^3 bits
- Synaptic Weights (e.g. 8/12/16/32 bits): N^3 values = From N^3 x weight size bits

For example, in a 150 x 150 network with a 32 bits precision, the size required is 1503 x 32 = 102.9 Mb. Fig. 3 illustrates the memory required to allocate the synapses weights of the proposed algorithm with different data sizes assuming configurations that contain the same number of layers and neurons per layer.

Table III. Spiking neural network profiling

Processing block	*Execution time*	*Number of iterations*	*Multiplications per neuron*	*Add/sub per neuron*
Update synapses per neuron	2.15%	n_{total}	1	1
Get total synaptic current (I_{syn})	90.20%	$n_{total}\ x\ n_{layer}$	n_{layer}	1
Izhikevich's equations per neuron	7.65%	n_{total}	14	17
Total	100%	$n_{total}\ x\ (2 + n_{layer})$	$n_{layer} + 15$	19

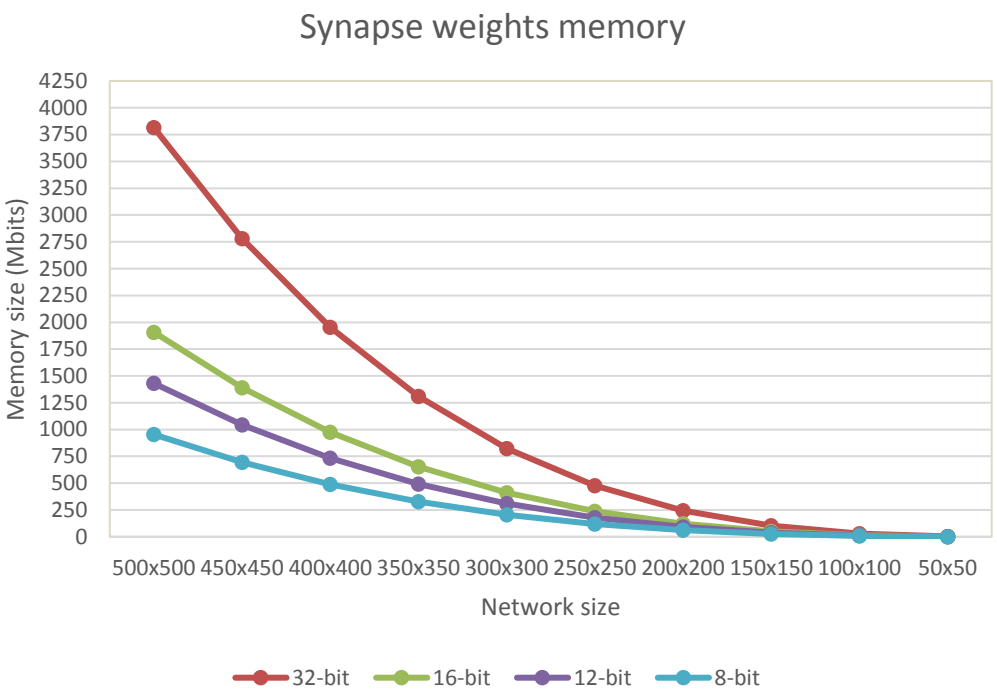

Fig. 3. Synapse weights memory required estimates

In the Zynq family the Zynq 7020 device has 4.9Mbits of BlockRAM which is also needed to support the DMA transfers and buffer data among other logic that must be implemented in the device. The point in Fig.3 corresponding to the 80x80 network and 8 bit precision needs 3.9 Mbits of memory and, realistically, will be the largest possible network with a total of 6,400 neurons. A larger Zynq 7100 with 26.5 Mbits of memory will be limited to approximately 150x150 or 22,500 neurons. This requirement creates a significant limitation and to overcome this problem alternative solutions are explored based on synaptic weight streaming in the following section.

4. DATA STREAMING AND PRECISION OPTIMIZATION

In the proposed solution synaptic weights are not stored in internal BlockRAMs but they are streamed from external memory when they are required. They are packed into 32 or 64-bit data streams, hence, the optimal data precisions that avoids additional decoding logic or wasted bits during packaging are 32-bit floating point or fixed-point of 32, 16 and 8-bits. Fig. 4 analyzes the impact of the different synaptic weight precision in the firing rates and the cross correlation of the fixed-point implementations against the floating point (as target). The graph plots the average, maximum and minimum firing rate of the neurons in the output layer of a $30x30$ fully connected feed-forward network of excitatory neurons with random weights with values between $2.5\,x\,10^{-2}$ and $5\,x\,10^{-4}$ along with the firing rate cross correlation against the implementation with the highest precision (floating point 32-bit).

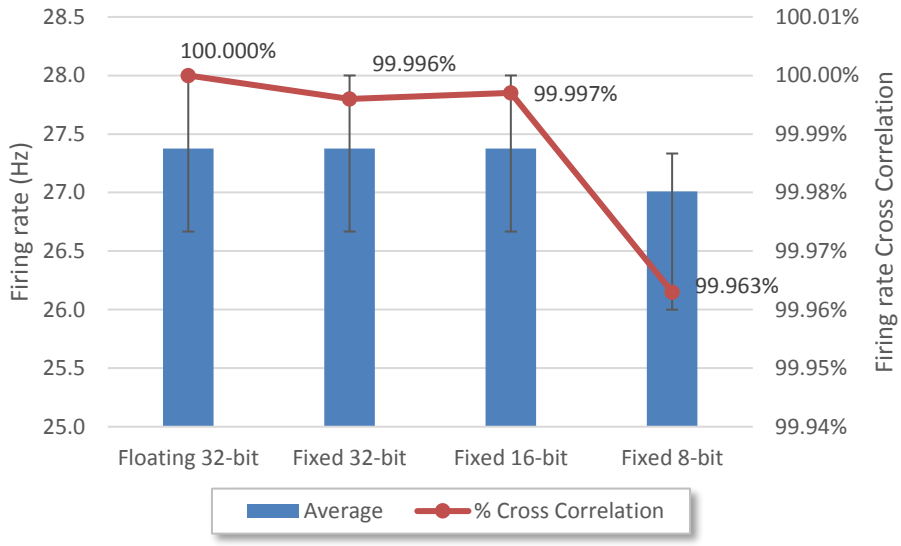

Fig. 4. Firing rate statistics over floating and fixed point precision weight's data types

The black lines and red dots indicates the minimum, maximum and average of the firing rates using the same network configuration with the four different implementations and those are given in Hertz. The blue columns (right axis) show the computed cross correlation percentages based on the firing rates using the same random weights of the implementations described previously and using the sample cross correlation implemented by MATLAB. It can be observed that in the fixed-point implementations of 32 and 16-bit the firing rate correlation error is almost null (less than 0.004%), whilst the error in the 8-bit implementation is larger at 0.04%, although accuracy is still higher than 99.9%. The precision of the rest of the variables that maintain the membrane potentials and synaptic currents are maintained at 32-bit since our experimentation has shown that the network is much more sensitive to these parameters. To achieve the transmission of the synapse weights on-demand without storing them in the FPGA memory the HLS processing block can receive up to four FIFO streams in parallel from external memory, matching the four high performance AXI ports available in the Zynq family. Each port is configured with a 64-bit width hence 256 bits of synaptic weights are available to be processed in parallel per cycle. An illustrative example of the streaming approach with a 100x100 network and 8-bit synapse weights is shown in Fig. 5.

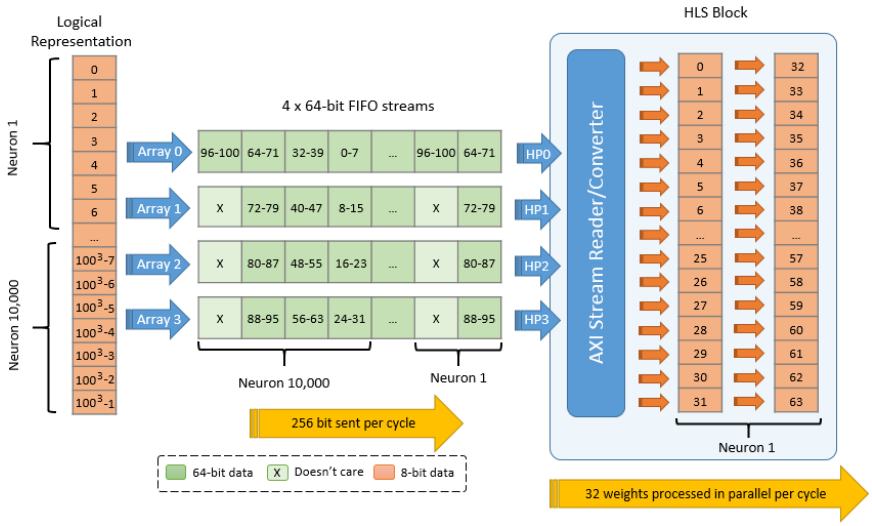

Fig. 5. Synapses weight streaming approach

Obtaining the total synaptic current and computing the Izhikevich model represent the most critical sections accordingly to Table III with around 90.20% and 7.65% of total complexity respectively. The combined computation involves at least $n_{layer} + 14$ multiplications and 18 additions per each processed neuron as can be obtained adding up the corresponding values in Table III. The main bottleneck lies in the reading throughput of the synaptic weights needed in order to get the conductance; therefore, as the maximum read throughput achieved by using the four high performance AXI ports available in the Zynq device is 256 bits per cycle the block latency for this block can be expressed as in eq. (6).

$$Pipeline\ interval\ per\ neuron = roundup\left(\frac{n_{layer} \cdot w_{bits}}{256}\right) \tag{6}$$

$$Block\ latency = n_{total} \cdot\ Pipeline\ interval\ per\ neuron +\ Iteration\ latency \tag{7}$$

The iteration latency is the latency required to process the accumulated conductance and the Izhikevich's equations for each pipelined neuron. It measures how many clock cycles the circuit needs to produce the first output. The Vivado HLS tool achieves an iteration latency after synthesis in the range of tens and the number of neurons n_{total} is in the range of thousands. For that reason the iteration latency may be omitted from eq. (7) and the block latency can be approximated as follows in eq. (8):

$$Block\ latency \approx n_{total} \cdot\ Pipeline\ interval\ per\ neuron \tag{8}$$

Then replacing the pipeline interval per neuron with eq. (6) we get eq. (9)

$$Overall\ latency \approx n_{total} \cdot \left(roundup\left(\frac{n_{layer} \cdot w_{bits}}{256}\right)\right) \tag{9}$$

For example, in a $100x100$ network with 8-bit synaptic weights, the latency of the block is approximately 40,000 clock cycles or 0.4 ms. Fig 6 applies this equation for different number of AXI ports, weight precisions and network sizes and compares it with the

performance that will be obtained if all the synaptic data could be stored in internal BRAMs. The BRAM performance estimation is based on the performance estimates obtained from Vivado HLS directly. Fig 6 shows that the performance of the 8-bit streaming approach with 4 ports shown in green and BRAM version shown in red are comparable up to network sizes of 300x300 and after that the streaming approach still offers competitive performance although it is not as good as storing all the data in the device BRAMs.

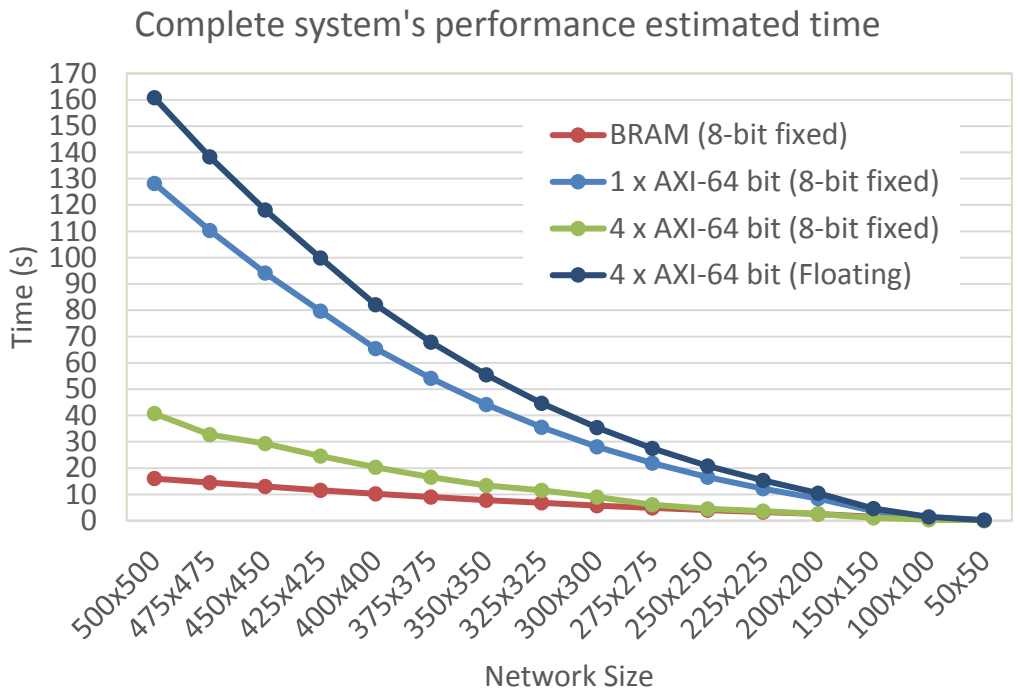

Fig. 6. Performance estimates for streaming and local synaptic weights

5. COMPLEXITY AND PERFORMANCE

The system is implemented in the Zynq family using the 8-bit precision configuration for synaptic weights and the PS and PL blocks are interfaced as shown in Fig 7 with an additional AXI DMA IP. Fig. 7 shows that the 4 available HP (high-performance) ports are used to stream data in and out of the device plus one GP (general purpose) point for controlling purposes. The results in terms of device utilization are depicted in Fig. 8 that mean that the 500*x*500 (i.e. 250K neurons) network will be suitable for the largest Zynq 7100 using 81% of internal BRAM resources.

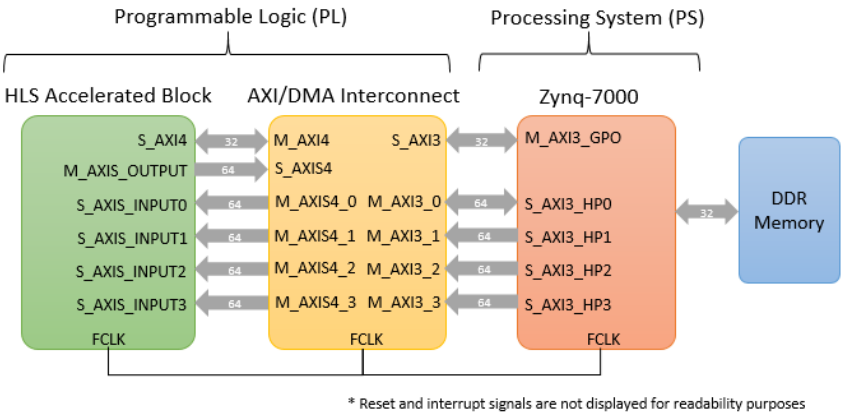

Fig. 7. System block diagram

A smaller device such as the Zynq 7020 available in the zc702 board limits the network complexity to approximately 170x170. For this configuration size the limiting factor are the DSP blocks since 213 out of 220 available are used.

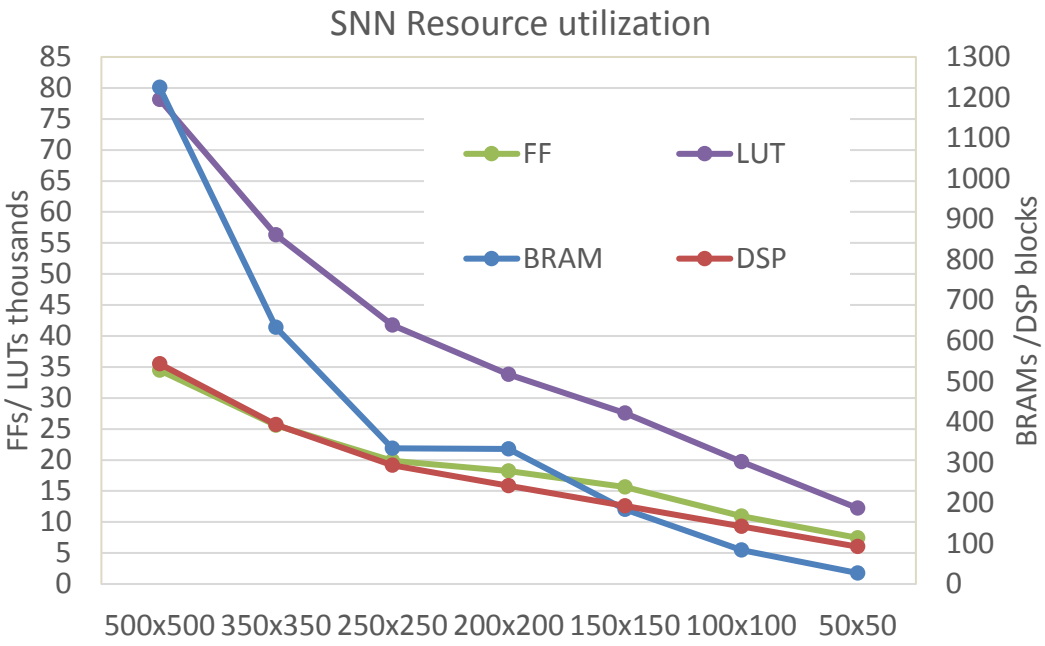

Fig. 8. Complexity analysis for resource utilization

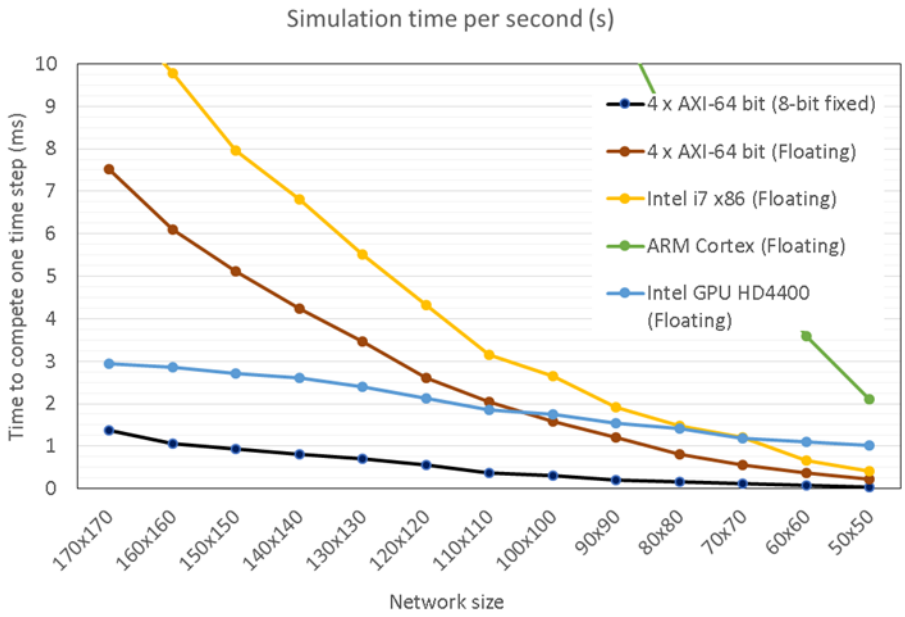

Fig. 9. Performance results

To measure and compare performance we use the zc702 board which is available for this research and consider sizes ranging from 50x50 to the maximun possible of 170x170 network. Fig. 9 compares the performance of the S2NN network on the Zynq device at 150 MHz with 8-bit and floating point precissions against equivalent OpenCL version mapped to the Core i7-4510U CPU @ 2.00GHz and the embedded Intel GPU HD4400. A serial version mapped to the ARM Cortex A9 device present in the Zynq device is also presented as a reference. The figure confirms that the 8-bit fixed point version offers higher throughput and can maintain real-time performance up to a network size of 150*x*150 neurons with a 150 MHz clock.

6. ENERGY EFFICIENCY AND PROPORTIONALITY

In the proposed system the active area in the S2NN network is configurable at run-time so that the number of layers and neurons per layer can be varied, and the neurons that are not part of the active area do not get computed. In a practical application this could mean that depending on the application the size of the network could change and adapt the number of active neurons to the complexity of the task. Assuming that the objective is to maintain real-time performance so that emulation time and real time are equivalent, a 150*x*150 network will need to run at 150 MHz as shown in Fig. 9 but the same network with an active area of 50*x*50 will only need to run at a fraction of that frequency to maintain real-time performance. In this section we consider three networks with 50*x*50, 100*x*100 and 150*x*150 neurons and investigate how voltage and frequency can be adapted to maintain real-time performance with minimum energy. Adaptation of voltage and frequencies is achieved using the voltage regulators and mixed mode frequency generation using a power adaptive architecture presented in Fig.10. This architecture is part of our previous work [Nunez-Yanez et al. 2016] and the interested reader is refer to that paper for more information. It consist of a DVS (Dynamic Voltage Scaling) unit that uses the PMBUS present in the device to adjust voltage levels and measure current and a DFS (Dynamic Frequency Scaler) that uses the MMCM (Mixed Mode Clock Managers) to synthesis new frequencies on the fly. The

user design is the S2NN in this paper and the AXI-PCAP port can be used to load different network configurations on the fly without altering the DFS and DVS logic and status. As shown in our previous work on DVFS [Nunez-Yanez et al. 2016] this DVFS-enable architecture has a logic overhead of around 5% which has a very small impact in power at the same voltage and frequency because in the FPGA device most of the power is not due to the additional configured logic cells but due to leakage and clock networks.

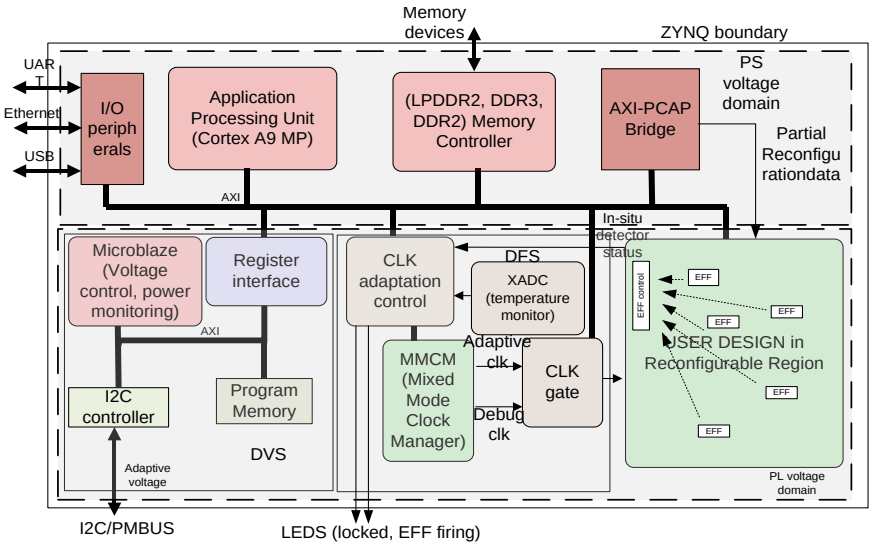

Fig. 10 Power adaptive system architecture

Power is measured in active (e.g. network running) and idle (e.g. network idling) states in Fig. 11 and Fig. 12 respectively. The power measurements include both static and dynamic power. The points in the X axis indicate the FPGA voltage, the FPGA frequency, the DDR3 memory voltage and the DDR memory frequency. For example the 150*x*150 configuration point 1.0_150_1.5_533 indicates that the FPGA is at nominal voltage of 1.0 volts and frequency of 150 MHz while the DDR3 memory is at nominal voltage of 1.5 volts and 533 MHz. For the 100*x*100 network there are three sets of bars with three bars each. The left set of three bars show the power usage at nominal voltage, the middle bars shows the power usage when frequency has been scaled down to 60 MHz that is the mininum frequency required to maintain real-time performance for this network size. Then the right set of bars show the effect of scaling DDR3 frequency to 0.9 volts and 200 MHz which provides enough memory bandwith to maintain the real-time performance of the network at 60 MHz. The reduction of active power with voltage and frequency scaling is 64% and 84% for the FPGA and DDR devices respectively for the 100*x*100 SNN. The same approach can be applied to the 50**x**50 network, although in this case only a 9 MHz clock is needed in the FPGA to support real-time performance. The reduction of power in this case is 88% and 84% for the FPGA and DDR devices respectively. The figures show that the most power intensive component is the DDR memory which is significantly higher than the FPGA active power and this is the case for all configurations. The streaming approach used by S2NN means that a signicant amount of bandwith is required since the four available high-performance ports are used constantly to move synaptic weights to the accelerator. As the frequency of the accelerator reduces in the voltage-scaled

configurations, the bandwith requested to the memory controller also reduces and this means that it is possible to reduce its own clock frequency without affecting overall performance.

To calculate the energy results shown in Fig. 13 we consider that the smaller configurations when running at the maximum frequency of 150 MHz will complete its computation early and they will need to wait for the next simulation step in idle state. The total energy is obtained as the addition of the active energy and the idle energy. For example, the 150*x*150 network does not have idle time and the only component is active energy. Similarly the voltage and frequency scaled configurations removed the idle energy since they remain active during the whole simulation step.

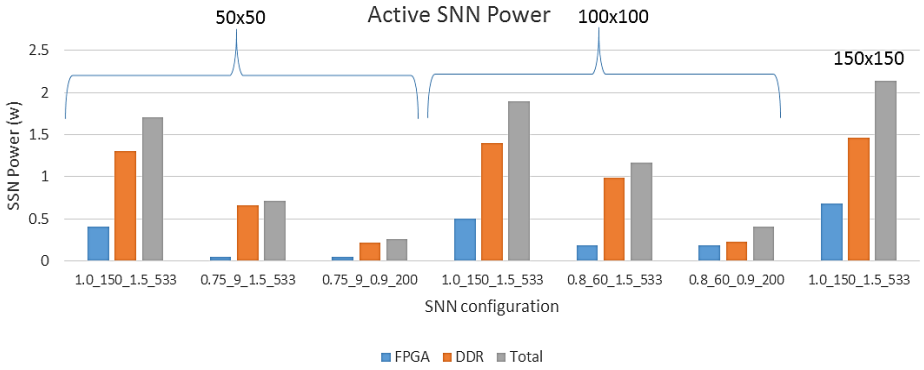

Fig. 11. S2NN FPGA and DDR active power

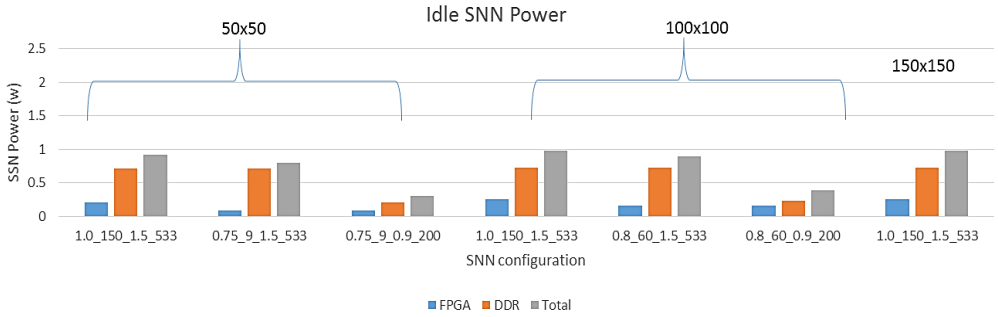

Fig. 12. S2NN FPGA and DDR idle power

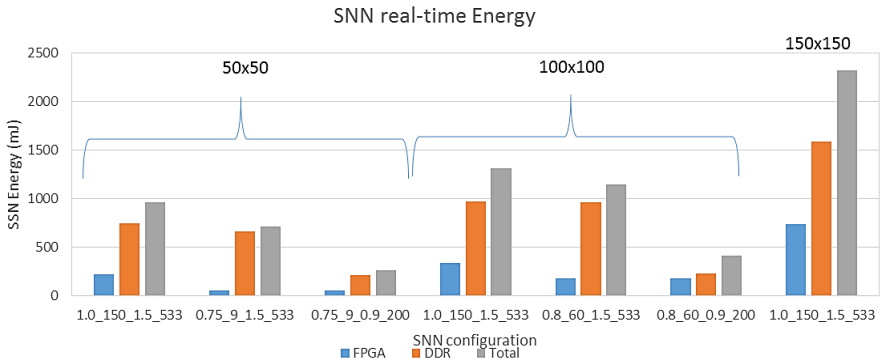

Fig. 13. S2NN FPGA and DDR energy

The results of Fig. 13 measure the energy required to perform one second of network activity and confirm the benefits of voltage scaling to reduce the energy requirements of the FPGA and memory devices. The 50*x*50 network benefits from an energy reduction of 76% and 69% for the FPGA and DDR3 memory respectively. The energy reduction for the 100*x*100 network is 48% and 77% for the FPGA and DDR3 memory respectively. For both cases the DDR3 switches between two fixed working points with frequency dropping to 200 MHz at 0.9 Volts. Even lower values of frequency in DDR3 memory will be enough to maintain performance for the smaller network configurations, but 200 MHz is the lower value accepted by the memory controller and 0.9 the lowest stable voltage. On the other hand, the FPGA has a wider range of working points and drops to 9 MHz and 60 MHz in each of the cases and this explains why the energy reductions are different in the FPGA device for each configuration. The FPGA frequency is generated by the internal mixed mode clock managers (MMCM) available in the FPGA device and the range of possible frequencies is much higher than that possible during the memory controller configuration.

Overall the measured results indicate that to perform 1 second of network activity costs 2,300 mJ , 413 mJ and 263 mJ for the 150*x*150, 100*x*100 and 50*x*50 S2NN configurations. As a reference point the same approach has been followed measuring the power in an Intel architecture considering active and idle values. An Intel Core I7 4770K at 3.5 GHz active and 0.8 GHz idle uses 119,600 mJ (only active since takes longer than one second), 37,000 mJ (36,000 mJ active + 1,000 mJ idle) and 14,100 mJ (5,600 mJ active + 8,500 mJ idle) for each of the network sizes with an average active power consumption of 40 Watts and idle power of 10 Watts. To obtain this energy value, the power in the CPU has been estimated using Intel software that allows access to the processor energy counters and only the processor package energy is considered for the Intel case (not main memory as done in the Zynq device). The intel i7 approach is the solution with the algorithm executed only by the CPU (traditional approach), it doesn't include GPU execution thus it doesn't include GPU consumption. The times measured for one second of a real time execution in the Intel platform are 2.99 s, 0.9 s and 0.15 s. In other words, only the two smaller network configurations can achieve a real time simulation. The estimation shows that the Zynq implementation uses approximately less than 2% of the Intel CPU energy.

7. CONCLUSIONS

This paper has presented a high-performance streaming spiking neural network called S2NN designed using C++ and mapped to the FPGA device with the Vivado high-level synthesis and implementation tools. The feature of being able to activate different sections of the network is exploited to obtain an energy proportional system in which the voltage and frequency of the processing and memory components are scaled to the levels required to maintain real-time performance. The measurements confirm that the DDR memory is more power and energy intensive than the CPU and programmable logic resources and that the standard memory controller and DDR3 devices present in the ZC702 board are able to perform memory power management with voltage scaling. Currently the CPU present in the Zynq device is used only to move network configuration data, activate and collect results from the FPGA side but our current work focuses on using the CPU side to deploy different learning algorithms that modify the synaptic weights and neuron configuration to perform useful functions. This embedded solution could be suitable to portable applications working in energy constrained scenarios such as automotive and robotics; and so, complements the currently available large scale brain emulators. Scaling the network to more than one device will be possible as long as each device has access to the weights of the part of the network it implements so streaming of weight information is local to the device and the output spikes written by one device in each time step are read by the next device. We have made the S2NN IP including source code and integration blocks available open-source at http://seis.bris.ac.uk/~eejlny/downloads/s2nn_package.zip to encourage reproducible results and further work in this field.

REFERENCES

E. Izhikevich, “Which Model to Use for Cortical Spiking Neurons?,” IEEE Trans. Neural Netw. IEEE Transactions on Neural Networks, vol. 15, no. 5, pp. 1063–1070, 2004.

Luiz André Barroso and Urs Hölzle. 2007. The Case for Energy-Proportional Computing. *Computer*40, 12 (December 2007), 33-37. DOI=http://dx.doi.org/10.1109/MC.2007.443

M. Hosseinabady and J. L. Nunez-Yanez, "Energy optimization of FPGA-based stream-oriented computing with power gating," 2015 25th International Conference on Field Programmable Logic and Applications (FPL), London, 2015, pp. 1-6. doi: 10.1109/FPL.2015.7293946

Howard David, Chris Fallin, Eugene Gorbatov, Ulf R. Hanebutte, and Onur Mutlu. 2011. Memory power management via dynamic voltage/frequency scaling. In Proceedings of the 8th ACM international conference on Autonomic computing (ICAC '11). ACM, New York, NY, USA, 31-40. DOI=http://dx.doi.org/10.1145/1998582.1998590

F. Akopyan et al., "TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, no. 10, pp. 1537-1557, Oct. 2015. doi: 10.1109/TCAD.2015.2474396

S. B. Furber, F. Galluppi, S. Temple and L. A. Plana, "The SpiNNaker Project," in Proceedings of the IEEE, vol. 102, no. 5, pp. 652-665, May 2014. doi: 10.1109/JPROC.2014.2304638

K. Cheung, R. Schultz and P. H. W. Leong, "A parallel spiking neural network simulator," Field-Programmable Technology, 2009. FPT 2009. International Conference on, Sydney, NSW, 2009, pp. 247-254. doi: 10.1109/FPT.2009.5377667

M. Ambroise, T. Levi, Y. Bornat, and S. Saïghi , “Biorealistic spiking neural network on FPGA,” 2013 47th Annual Conference on Information Sciences and Systems (CISS), 2013.

K. L. Rice, M. A. Bhuiyan, T. M. Taha, C. N. Vutsinas, and M. C. Smith, “FPGA Implementation of Izhikevich Spiking Neural Networks for Character Recognition,” 2009 International Conference on Reconfigurable Computing and FPGAs, 2009.

M. A. Bhuiyan, A. Nallamuthu, M. C. Smith, and V. K. Pallipuram, “Optimization and performance study of large-scale biological networks for reconfigurable computing,” 2010 Fourth International Workshop On High-Performance Reconfigurable Computing Technology And Applications (Hprcta), 2010.

F. Naveros, N. R. Luque, J. A. Garrido, R. R. Carrillo, M. Anguita, and E. Ros, “A Spiking Neural Simulator Integrating Event-Driven and Time-Driven Computation Schemes Using Parallel CPU-GPU Co-Processing: A Case Study,” IEEE Trans. Neural Netw. Learning Syst. IEEE Transactions on Neural

Networks and Learning Systems, vol. 26, no. 7, pp. 1567–1574, 2015.

D. Thomas and W. Luk, “FPGA Accelerated Simulation of Biologically Plausible Spiking Neural Networks,” 2009 17th IEEE Symposium on Field Programmable Custom Computing Machines, 2009.

K. Cheung, R. Schultz, W. Luk, “A large scale spiking neural network accelerator for FPGA systems” in 22nd International Conference on Artificial Neural Networks, Lausanne, Switzerland, September 11-14, 2012, pp. 113-120, DOI=http://dx.doi.org/10.1007/978-3-642-33269-2_15

S. W. Moore, P. J. Fox, S. J. T. Marsh, A. T. Markettos and A. Mujumdar, "Bluehive - A field-programable custom computing machine for extreme-scale real-time neural network simulation," Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on, Toronto, ON, 2012, pp. 133-140. doi: 10.1109/FCCM.2012.32

G. Smaragdos, S. Isaza, M. F. V. Eijk, I. Sourdis, and C. Strydis, “FPGA-based biophysically-meaningful modeling of olivocerebellar neurons,” Proceedings of the 2014 ACM/SIGDA international symposium on Field-programmable gate arrays - FPGA '14, 2014.

Juan Carlos Moctezuma, Joseph P. McGeehan, Jose Luis Nunez-Yanez, Biologically compatible neural networks with reconfigurable hardware, Microprocessors and Microsystems, Volume 39, Issue 8, November 2015, Pages 693-703

J. Luis Nunez-Yanez, M. Hosseinabady and A. Beldachi, "Energy Optimization in Commercial FPGAs with Voltage, Frequency and Logic Scaling," in IEEE Transactions on Computers, vol. 65, no. 5, pp. 1484-1493, May 1 2016. doi: 10.1109/TC.2015.2435771

Dmitri Vainbrand and Ran Ginosar. 2011. Scalable network-on-chip architecture for configurable neural networks. Microprocessors and Microsystems, Volume 35, Issue 2 , March 2011, Pages 152-166.

S. Pande, F. Morgan, S. Cawley, T. Bruintjes, G. Smit, B. McGinley, S. Carrillo, J. Harkin, and L. McDaid, “Modular neural tile architecture for compact embedded hardware spiking neural network,” Neural Process. Lett., vol. 38, no. 2, pp. 131–153, Jan. 2013.